Namal University, Mianwali

Department of Electrical Engineering

Communication Systems (Lab)

Lab – 1

MATLAB Review – Basic Operations and Functions

Student Name	Student ID
Fahim Ur Rehman Shah	NIM-BSEE-2021-21

Instructor: Dr. Sajjad Ur Rehman

Lab Engineer: Faizan Ahmad

Version 1.0 – 27/03/2021 – Engr. Arbaz
Version 1.1 – 27/03/2022 – Engr. Arbaz
Version 1.2 – 20/03/2023 – Engr. Imtiaz Ul Hassan
Version 1.3 – 26/02/2024 – Engr. Faizan Ahmad

Introduction

The purpose of this lab is to provide the students with a review of numerical computations software, MATLAB, as well as provide revision for some of the basic signals and systems concepts, which will be used in the coming labs.

Course Learning Outcomes

CLO2: Develop software simulations to observe the performance of analog and digital communication systems.

CLO4: Report desired results proofs and calculations.

Equipment

□ Software

- ○ MATLAB

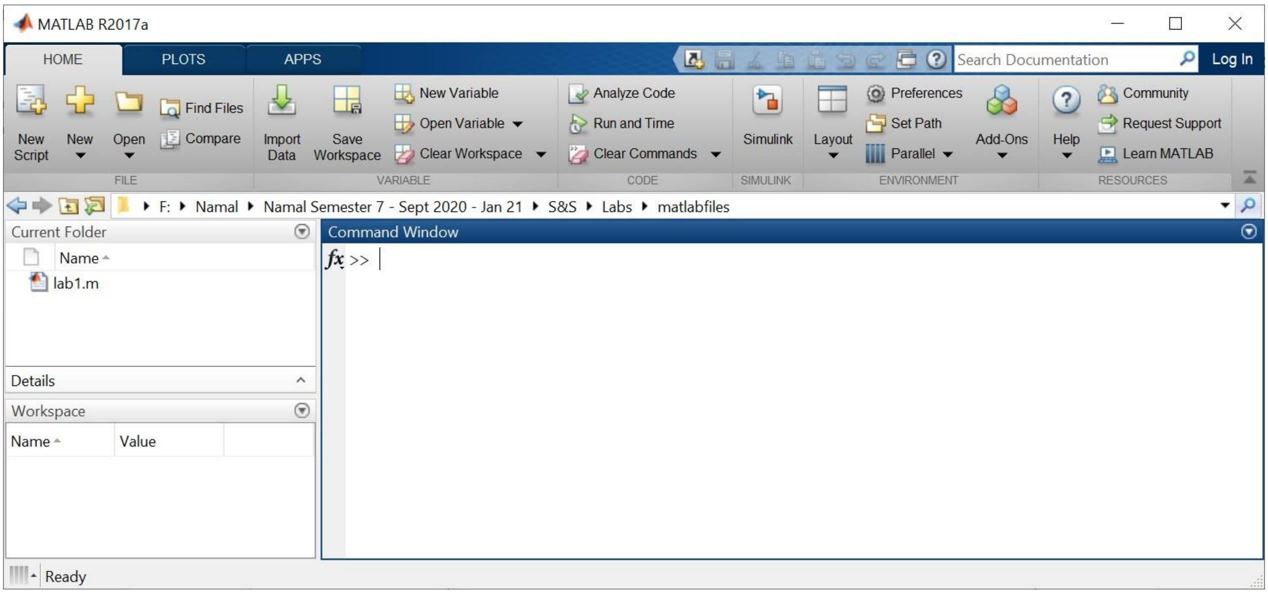
Instructions

- This is an individual lab. You will perform the tasks individually and submit the required files at the end of the lab.
- Plagiarism or any hint thereof will be dealt with strictly. Any incident where plagiarism is caught, both (or all) students involved will be given zero marks, regardless of who copied whom. Multiple such incidents will result in disciplinary action being taken.

Getting started with MATLAB

1. User Interface

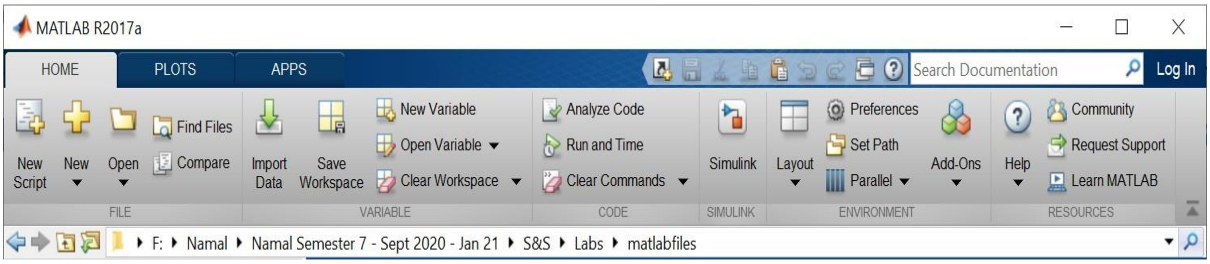
- Open MATLAB from either the desktop icon or by searching in the start menu. The following window should be displayed:



- As you can see, the main MATLAB window consists of several partitions that are the main toolbar, Command window, Workspace and Current Folder window.

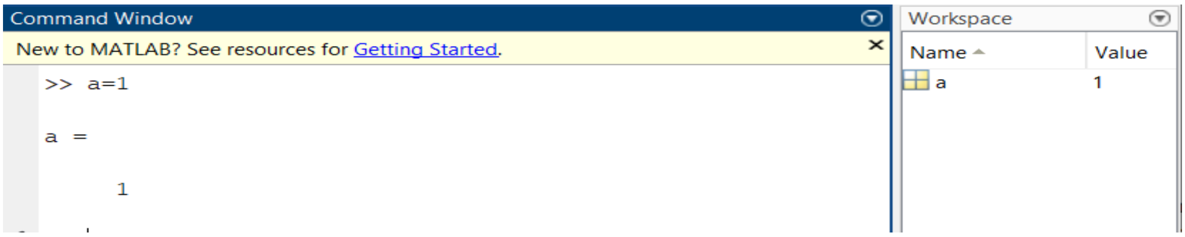
Menu Bar/Tool Bar

- The icons at the top of the window change with the change in tab selected. The bar at the bottom of this menu shows the current directory.



Command Window and Workspace:

- We type all our commands in this window at the prompt ‘ >> ’ and press Enter button from keyboard to see the results of our operations. In the command window if any statement is followed by a semicolon ‘ ; ’ result of this statement will not be shown.



```
>> a=1

a =

     1
```

Name	Value
a	1

Table 1.1: Basic commands of Workspace Window

Commands	**Function**
Who	List all the variables which are saved in workspace
Whos	Lists more information about each variable including size, bytes stored in the computer, and class type of the variables
clear all	Remove all variables and functions from the workspace
clear VAR1	Remove only VAR1 from the workspace
Clc	Clear the command window, but do not removes the variables from workspace

5. Editing Window

- Small computations can be performed directly in the command window, however a command once entered cannot be changed. Longer codes are usually written in the Editor, where they can be modified as well as saved as .m files. Go to New>Script or click on the New script icon in the toolbar to open a new editable file.
- The file must be saved before it can be run. MATLAB does not allow spaces in the file names, however underscore can be used in both file names and variable names in MATLAB. The MATLAB files are saved with the '.m' extension.
- When you run the program, if your program displays any output, it will be displayed in the Command window otherwise only a program execution notification appears in the window.
- Percentage (%) sign is used for comments in MATLAB.

6. Vectors and Matrices

Till now, we have been storing single values using variables. We can also store a sequence of values using structures called vectors or matrices. Vectors are one dimensional, while matrices are two dimensional storage spaces that store multiple values.

Vectors

- There are two kinds of vectors in MATLAB i.e. row vectors and column vectors. In a row vector, there is a single row and multiple columns while in a column vector, there is a single column and several rows.
- A row vector in MATLAB can be created by enclosing any numbers separated by spaces or commas in square brackets, e.g.,

```
>> row_vector = [1 3 5]
row_vector
   =1  3   5
```

- A column vector can be created by simply putting semi colon (;) after each number, e.g.,

```
>> column_vector = [2;4;6]
column_vector =
   2
   4
   6
```

Basic Matrix Operation:

- Suppose A and B are two matrices as defined below;

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \xrightarrow{MATLAB\ Command} \text{A = [1 2 3; 4 5 6; 7 8 9]}$$

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \xrightarrow{MATLAB\ Command} \text{B = [1 0 0; 0 1 0; 0 0 1]}$$

Table I: Basic Matrix Operations

Operation	**Operator**	**Example**	**Notes**
Plus	+	A+B	A and B must be of same dimensions
Minus	–	A–B	A and B must be of same dimensions
Multiply	*	A*B	A and B must be of compatible dimensions
Matrix Power	^	A^k	A must be square matrix, k must be a constant
Multiply (element by element)	.*	A.*B	Multiplies elements a_{ij} by b_{ij}

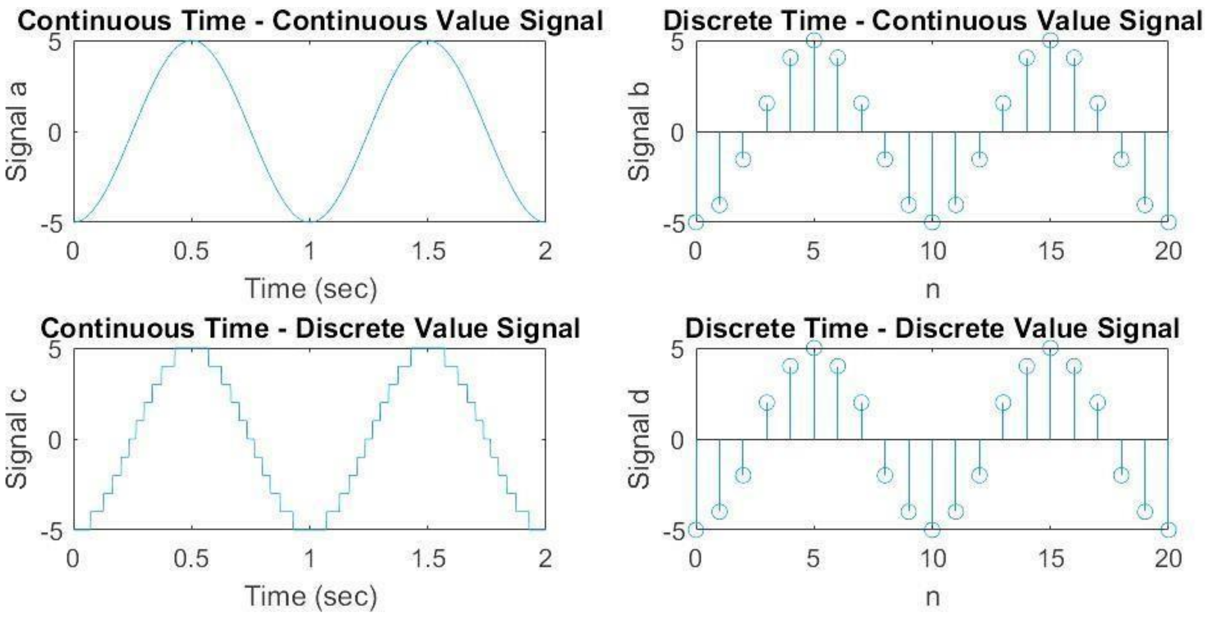
```
>> mat_1 = [1 2; 3 4];
>> mat_2 = [5 6; 7 8];
>> add_mat = mat_1 + mat_2
add_matrices =
         6    8
        10   12
```

Task 1 – Classification of Signals

We can divide the signals into many different categories, based on their properties. One such classification is based on whether the time and/or value axis is continuous or discrete. Based on this, the signals can be divided into four categories:

1. Continuous time – Continuous Value Signals
2. Discrete time – Continuous Value Signals
3. Continuous time – Discrete Value Signals
4. Discrete time – Discrete Value Signals

The first two are Analog signals, and the last two are Digital Signals.



Open a new .m file and save it as Lab1_yourUETno_yourname.m e.g. Lab1_123_Xyz.m. For future reference, all your .m files must be saved with the same name format.

Use `%%%%%%%%%%%%%% Exercise 1b %%%%%%%%%%%%%%%%%%%%%` as a separator in the same .m file and type the code for exercise 1b below it. For future reference, all your exercise parts / tasks must be separated in the same way.

Exercise 1

a) Run the following code and observe its output.

```
clear all;
clc;
%%%%%%%%%%%%%% Exercise 1a %%%%%%%%%%%%%%%%%%%%
t   = 0:0.001:2;
fa  = 1;    % frequency of a
Aa  = 5;     % amplitude of a
Pa  = 0;     % phase of a
A   = Aa*cos(2*pi*fa*t + pa);
```

```
figure(1)
subplot(2,2,1)
plot(t,a)
title('Continuous Time - Continuous Value
Signal');
xlabel('Time (sec)');
ylabel('Signal a');
```

Explain the working of this code.

b) Using the above code as reference, write a code to display a Discrete Time – Continuous Value sinusoidal signal b. Define a variable n instead of t, with value going from 1 to 20 with a step size of 1. Use frequency 0.1 for this signal, amplitude 5, and phase 0. Use the subplot command to choose a different part of the same figure window. Use stem instead of plot for a discrete time signal. Make sure to write appropriate title and labels.

c) Similar to part a and b, display a Continuous Time – Discrete Value sinusoidal signal c, with frequency 1, amplitude 5, and phase 0, in a different part of the same figure window. Note that this time the x axis will stay continuous, so plot will be used, but the values of c will have to be discretized, so that only integer values remain. You can use the 'round' function to discretize the values. Use MATLAB help to understand how it works. Make sure to write appropriate title and labels.

d) Display a Discrete Time – Discrete Value sinusoidal signal d, with frequency 0.1, amplitude 5, and phase 0, in a different part of the same figure window. Make sure to write appropriate title and labels.

Can a true continuous time or continuous value signal be defined in MATLAB? Why or why not?

What is the difference between plot and stem?

Task 2 – Continuous Time convolution

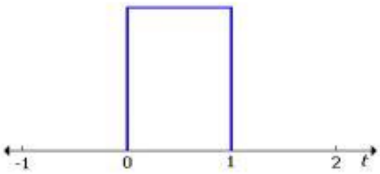
If both linearity and time-invariance hold, the output of the system can be found through a relation known as the Convolution Integral:

$$
\begin{aligned}
y(t) &= \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau \\
&= x(t) * h(t)
\end{aligned}
$$

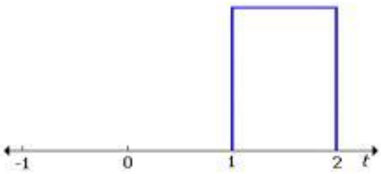
The above relationship means " x(t) convolved with h(t) " where h(t) is the impulse response of the LTI system. The variable t is taken to be a constant for the integration, which is done over the dummy

variable τ for all nonzero values of the input function. After integration the τ variable disappears, leaving a function of time which is in the output of the system.

Consider an example using the following system and input:



$x(t) = u(t) - u(t-1)$

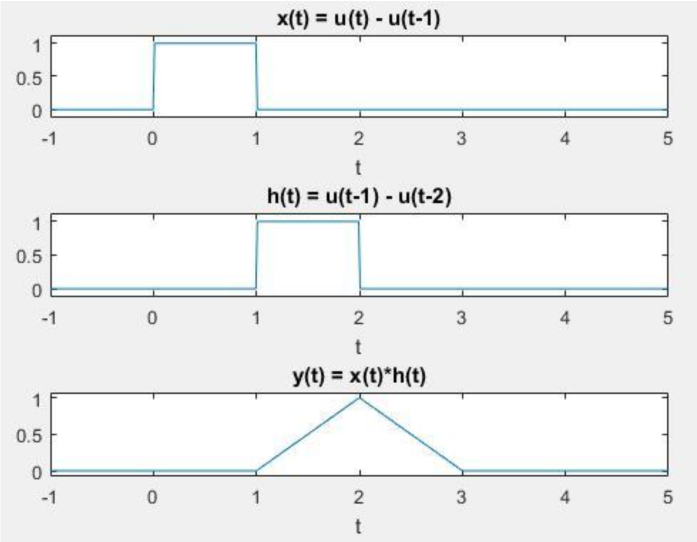


$h(t) = u(t-1) - u(t-2)$

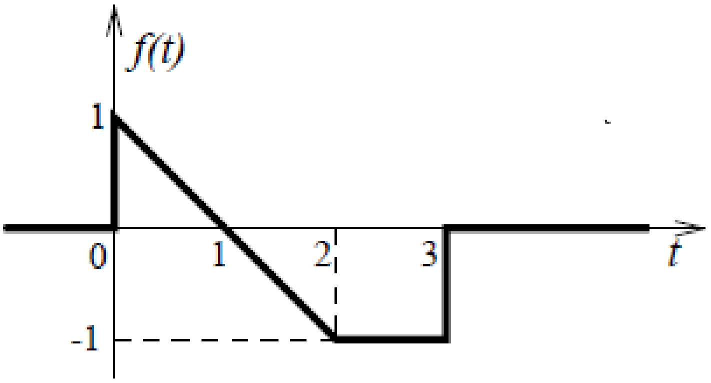
Matlab Implementation:

```
syms w u(t) x(t) r(t) h(t) y(t)
u(t) = heaviside(t);
x(t) = u(t)-u(t-1);
h(t) = u(t-1) - u(t-2);
y(t) = int(x(w)*h(t-w), w, -inf, inf);
subplot 311
ezplot(x(t), [-1 5])
title('x(t) = u(t) - u(t-1)')
subplot 312
ezplot(h(t), [-1 5])
title('h(t) = u(t-1) - u(t-2)')
subplot 313
ezplot(y, [-1 5])
title('y(t) = x(t)*h(t)')
```

Graph:



Piecewise Signal Generation:



Using Symbolic Tools (Continuous Time System):

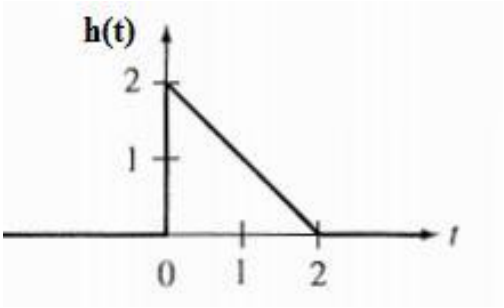
```
clear all;
clc
syms t g(t) u(t)
u(t) = heaviside(t)
f(t) = (-t+1).*(u(t)-u(t-2)) + (-1).*(u(t-2)-u(t-3))
ezplot(f(t),[-5 5])
grid on
```

<u>Excercise</u>

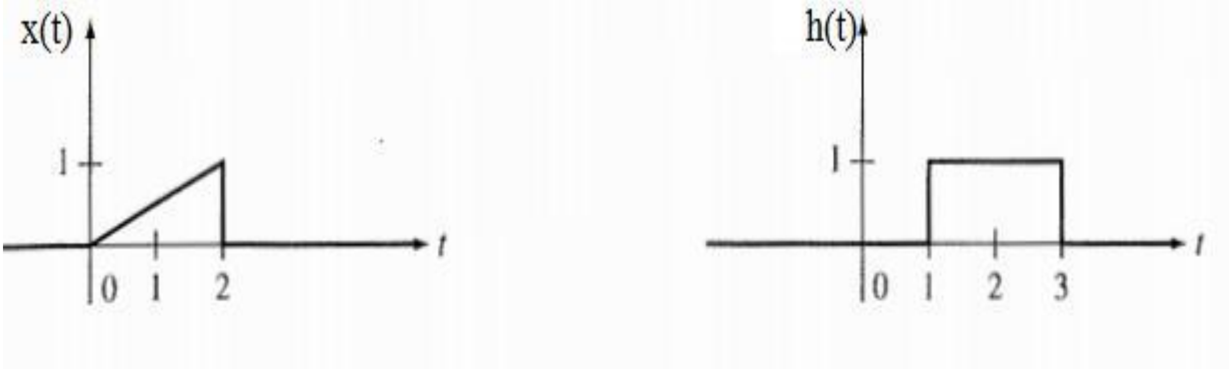
Convolve x(t) and h(t) to get y(t).

a)

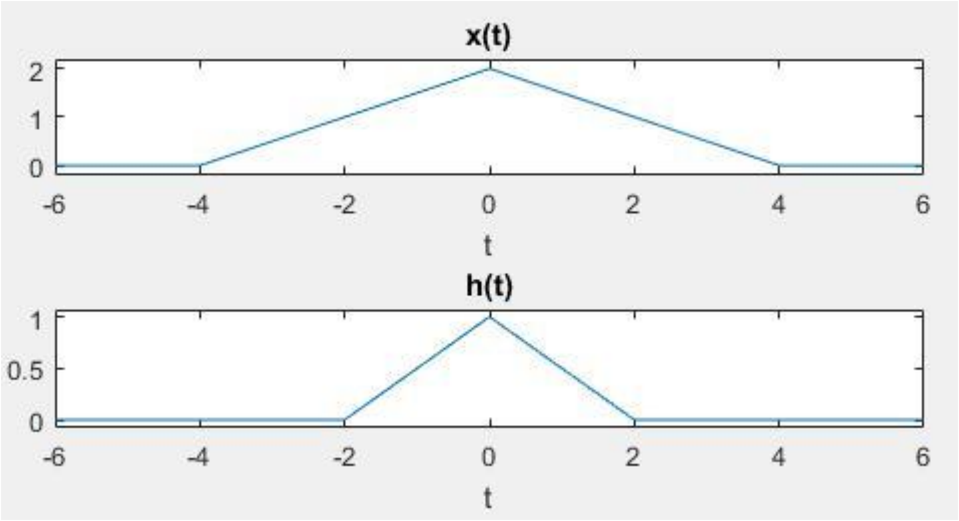
x(t) = u(t-2)



b)



c)



Report

For each exercise, include the code you write, explanation where asked, all outputs, as well as answers to any questions, as required. Upload your final report (containing general formalities like cover page containing your name and roll number etc.) to QOBE in the report submission folder.

Upload your final executable .m file as well in the separate submission folder on QOBE.

Exercise 1:

Part(a)

Code:

```
clear all;
clc;
t = 0:0.001:2;
fa = 1; % frequency of a
Aa = 5; % amplitude of a
pa = 0; % phase of a
A = Aa*cos(2*pi*fa*t + pa);
figure(1)
subplot(2,2,1)
plot(t,A)
title('Continuous Time - Continuous Value Signal');
xlabel('Time (sec)');
ylabel('Signal A');
```

Explaination:

This code generates a continuous-time signal \(A \) over the time interval from 0 to 2 seconds. The signal is a cosine wave with a frequency of 1 Hz (cycles per second), an amplitude of 5, and zero phase. The 'subplot' function divides the figure window into a 2x2 grid and selects the first subplot to plot the signal. The resulting plot shows how the amplitude of the cosine wave varies with time. This code snippet is helpful for visualizing continuous signals in MATLAB, commonly encountered in fields such as signal processing and control systems.

Part(b)

Code:

```
n = 0:1:20;
fa = 0.1; % frequency of a
Aa = 5; % amplitude of a
pa = 0; % phase of a
A = Aa*cos(2*pi*fa*n + pa);
figure(1)
subplot(2,2,2)
stem(n,A)
title('Discrete Time - Continuous Value Signal');
xlabel('Time (sec)');
ylabel('Signal A');
```

Explanation:

This MATLAB code generates a discrete-time signal \(A \) over the range of integers from 0 to 20. The signal is a cosine wave with a frequency of 0.1 cycles per sample, an amplitude of 5, and zero phase. The 'stem' function is used to plot discrete points of the signal at integer time instances. The resulting plot shows how the amplitude of the cosine wave varies at each discrete time point. This code snippet is useful for understanding and visualizing discrete signals in MATLAB, which are commonly encountered in digital signal processing and communication systems.

Part (C):

Code

```
clear all;
clc;
t = 0:0.001:2;
fa = 1; % frequency of a
Aa = 5; % amplitude of a
pa = 0; % phase of a
A= Aa*cos(2*pi*fa*t + pa);
Y=round(A)
figure(1)
subplot(2,2,3)
plot(t,Y)
title('Continuous Time - Discrete Value Signal');
xlabel('Time (sec)');
ylabel('Signal A');
```

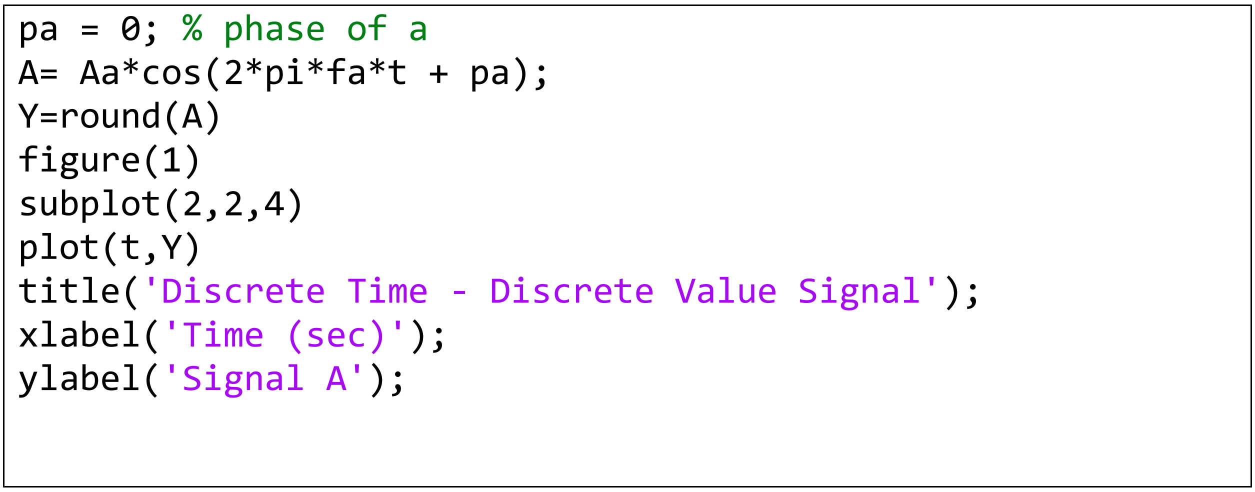
Explanation:

This MATLAB code generates a continuous-time sinusoidal signal \(A \) over the time interval from 0 to 2 seconds, then rounds each value to the nearest integer to create a discrete-value signal \(Y \). The 'subplot' function arranges the plots, and 'plot' visualizes the rounded signal over time. This process demonstrates how to convert a continuous signal into a discrete one by rounding its values, which can be useful for quantization in digital signal processing applications.

Part (C):

Code

```
clear all;
clc;
t = 0:1:20;
fa = 0.1; % frequency of a
Aa = 5; % amplitude of a
```



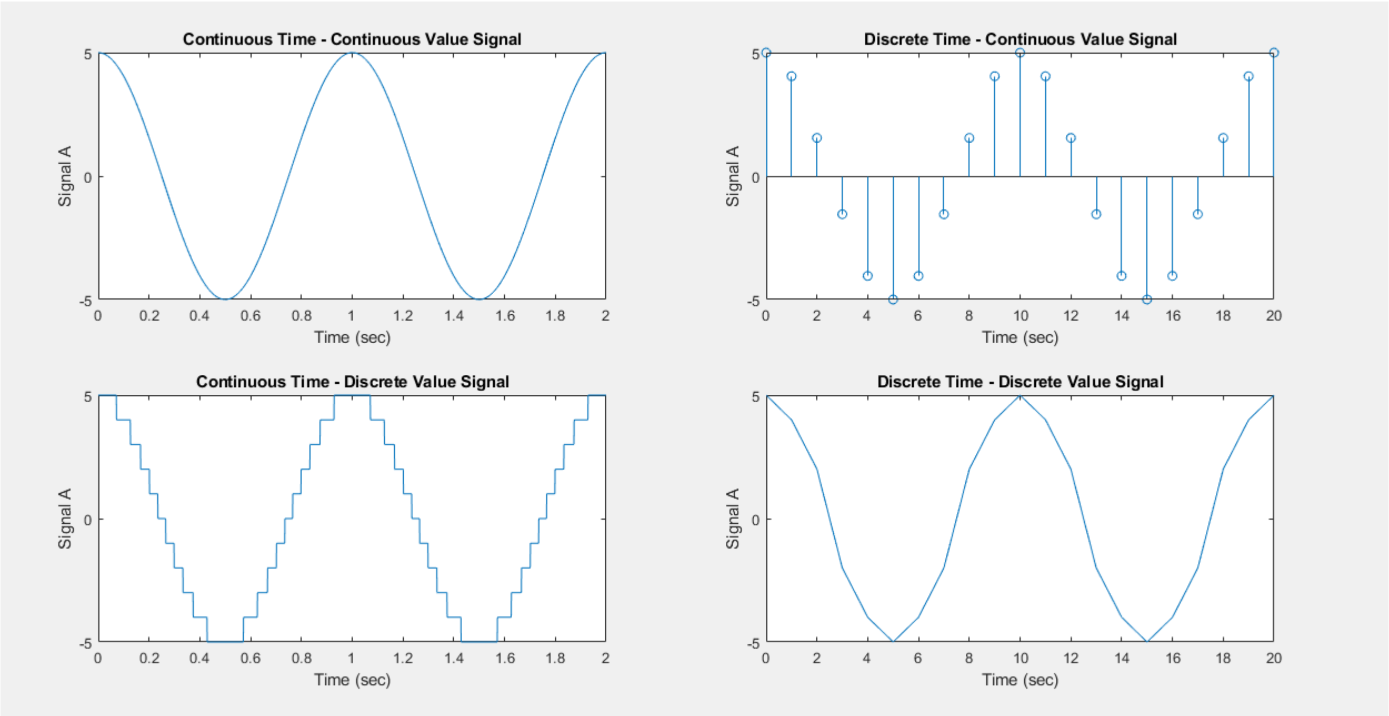
```
pa = 0; % phase of a
A= Aa*cos(2*pi*fa*t + pa);
Y=round(A)
figure(1)
subplot(2,2,4)
plot(t,Y)
title('Discrete Time - Discrete Value Signal');
xlabel('Time (sec)');
ylabel('Signal A');
```

Explanation:

This MATLAB code creates a discrete-time sinusoidal signal \(A \) over the time range from 0 to 20 seconds, then rounds each value to the nearest integer to produce a discrete-value signal \(Y \). The 'subplot' function arranges the plots, and 'plot' visualizes the rounded signal over discrete time instances. This process showcases how to convert a continuous sinusoidal signal into a discrete one with discrete values, which can be useful in digital signal processing applications where signals are represented digitally.

Exercise 1:

Output of a,b,c and d parts using subplot



Continuous Time - Continuous Value Signal

Discrete Time - Continuous Value Signal

Continuous Time - Discrete Value Signal

Discrete Time - Discrete Value Signal

- **Can a true continuous time or continuous value signal be defined in MATLAB? Why or why not?**

Ans:

In MATLAB, true continuous time or continuous value signals cannot be defined directly. This is because MATLAB operates in a digital environment where computations are performed on discrete data points. While MATLAB can simulate continuous-time behavior through techniques like using small time steps or interpolating between discrete points, it ultimately deals with discrete values. Real-world continuous signals are typically sampled and then processed in MATLAB as discrete-time signals. However, MATLAB provides functions and tools for working with discrete signals effectively, enabling simulations and analysis of continuous-time systems through techniques like numerical integration or Fourier analysis.

- **What is the difference between plot and stem?**
  - Plot draws a smooth line connecting points, good for showing trends or continuous data.
  - Stem shows individual points as markers on a vertical stem, useful for displaying discrete data or individual values without connecting lines.

Task 2 – Continuous Time convolution:

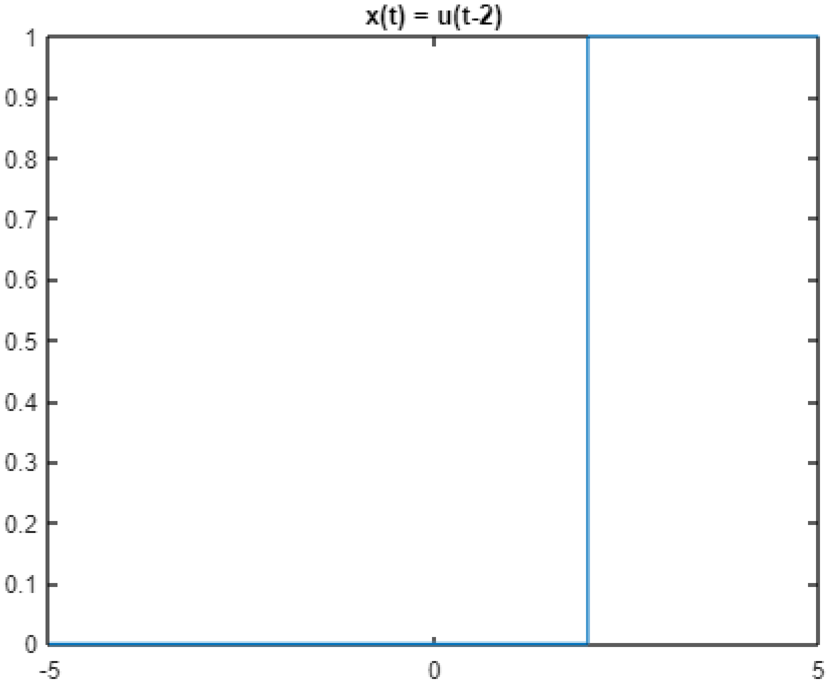
Exercise

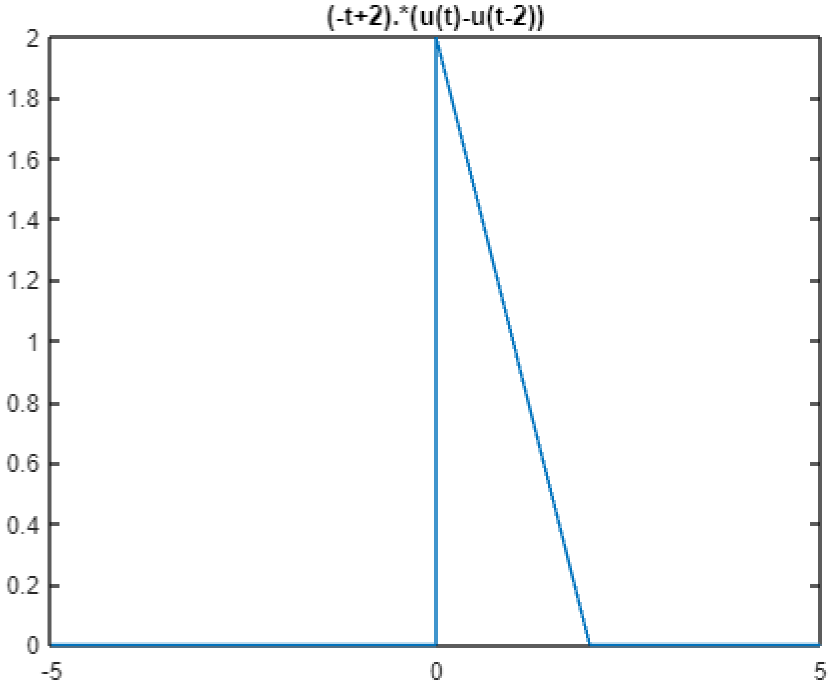
- ***a.***

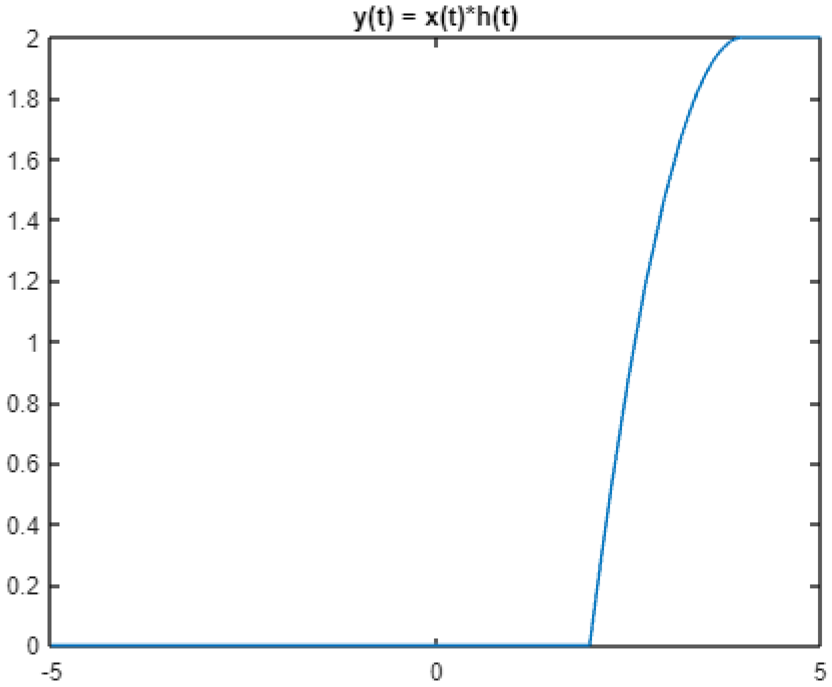
Code:

```
syms w u(t) x(t) r(t) h(t) y(t)
u(t) = heaviside(t);
x(t) = u(t-2);
h(t) =  (-t+2).*(u(t)-u(t-2))
y(t) = int(x(w)*h(t-w), w, -inf, inf);
fplot(x(t))
title('x(t) = u(t-2)')
fplot(h(t))
title('(-t+2).*(u(t)-u(t-2))')
fplot(y(t))
title('y(t) = x(t)*h(t)')
```

Output:







Explanation:

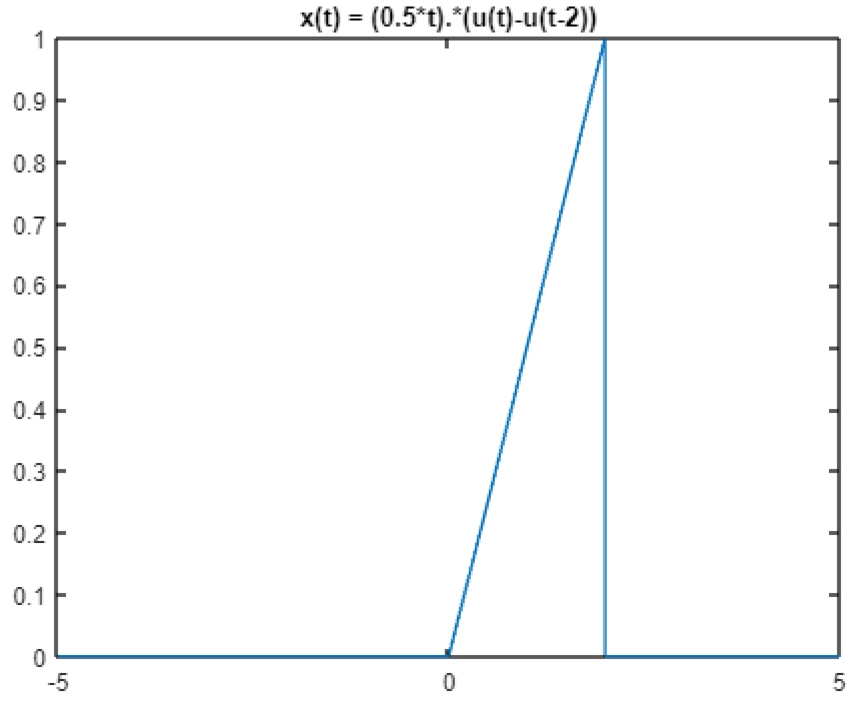
In this MATLAB code, we are defining symbolic functions and analyzing their behavior. We start by defining a step function u(t) that becomes 1 when t is greater than or equal to 0, and another function x(t) that represents a delayed step function by setting it to u(t-2). The function h(t) is defined as the product of (-t+2) and the difference of two step functions, creating a triangular pulse. Then, the convolution of x(t) and h(t) is calculated and stored in y(t). Finally, the code plots the graphs of x(t), h(t), and y(t) using the fplot function and provides titles for each plot. The first plot shows the delayed step function x(t), the second plot displays the triangular pulse h(t), and the third plot illustrates the convolution result y(t) of x(t) and h(t).

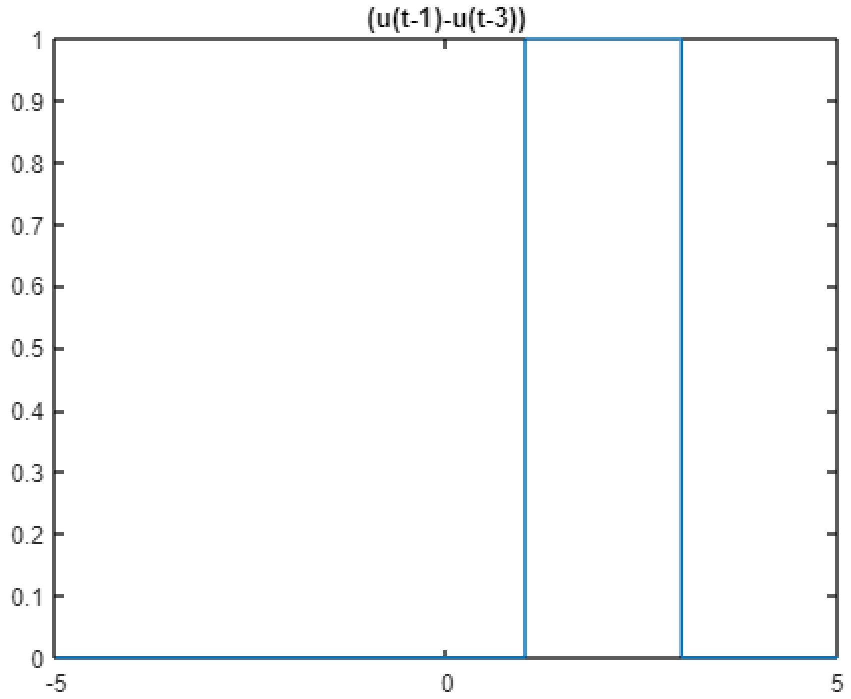
- ***b.***

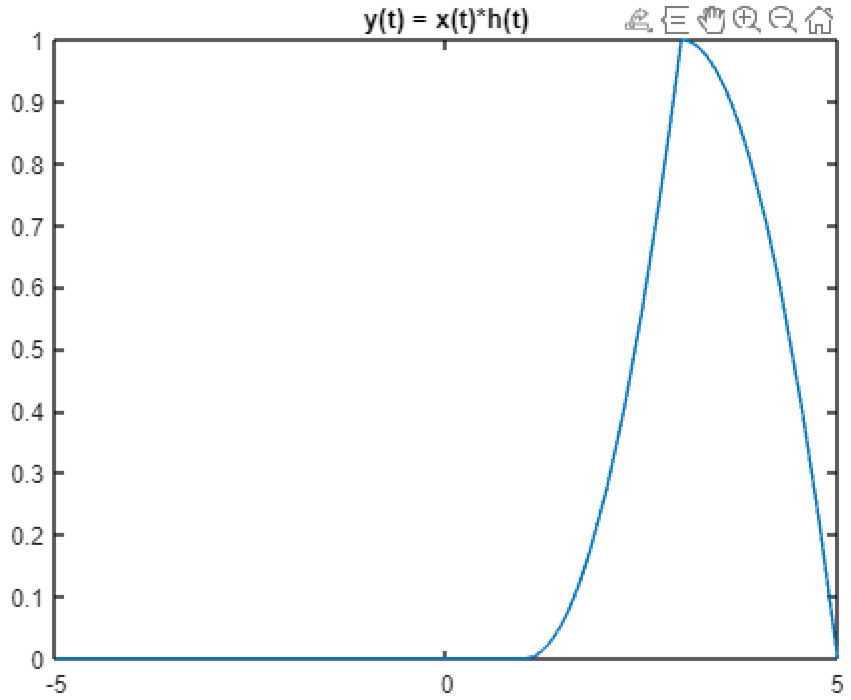
Code:

```
syms w u(t) x(t) r(t) h(t) y(t)
u(t) = heaviside(t);
x(t) = (0.5*t).*(u(t)-u(t-2));
h(t) =  (u(t-1)-u(t-3))
y(t) = int(x(w)*h(t-w), w, -inf, inf);
fplot(x(t))
title('x(t) = (0.5*t).*(u(t)-u(t-2))')
fplot(h(t))
title('(u(t-1)-u(t-3))')
fplot(y(t))
title('y(t) = x(t)*h(t)')
```

Output:







Explanation:

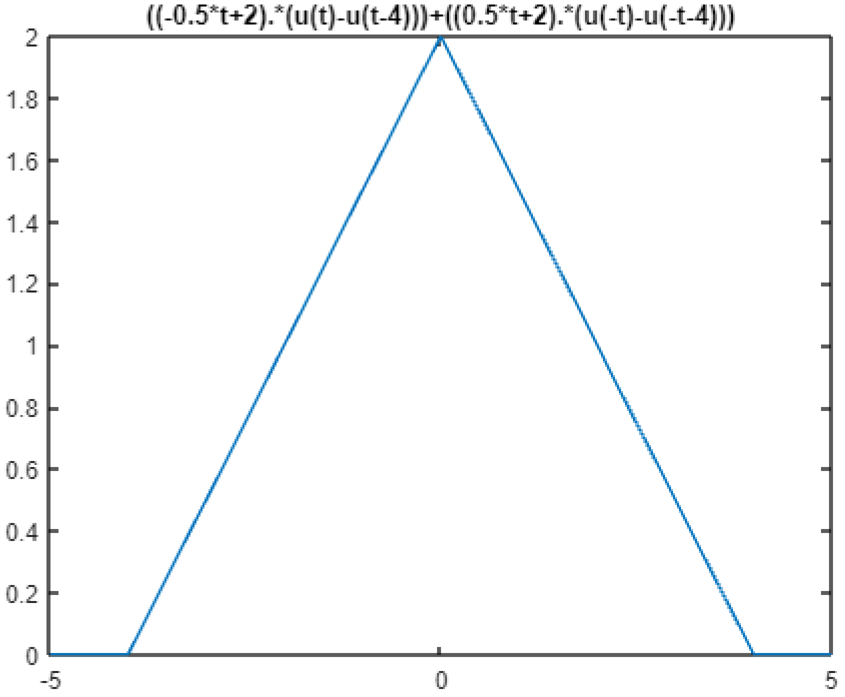
In this MATLAB code, symbolic functions are used to explore a signal processing scenario. The step function u(t) is defined, and then two other functions, x(t) and h(t), are created. The function x(t) represents a piecewise linear signal defined as 0.5*t for t between 0 and 2, and zero otherwise. The function h(t) is a rectangular pulse defined between t=1 and t=3. The code then calculates the convolution of x(t) and h(t), storing the result in y(t). The fplot function is used to graphically represent x(t), h(t), and y(t), each with their respective titles. The first plot depicts the piecewise linear signal x(t), the second plot illustrates the rectangular pulse h(t), and the third plot shows the convolution result y(t) of x(t) and h(t).

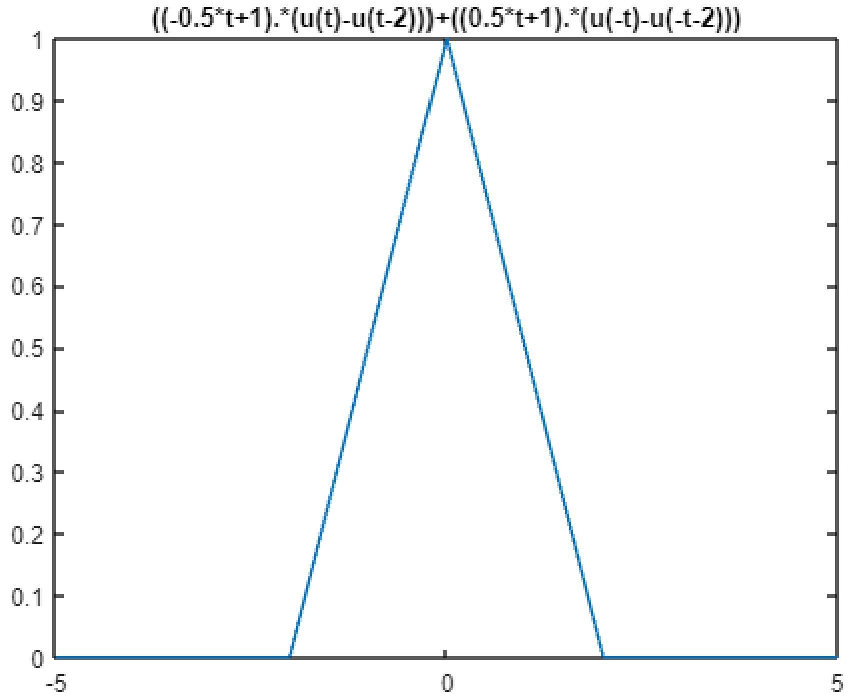
- ***c.***

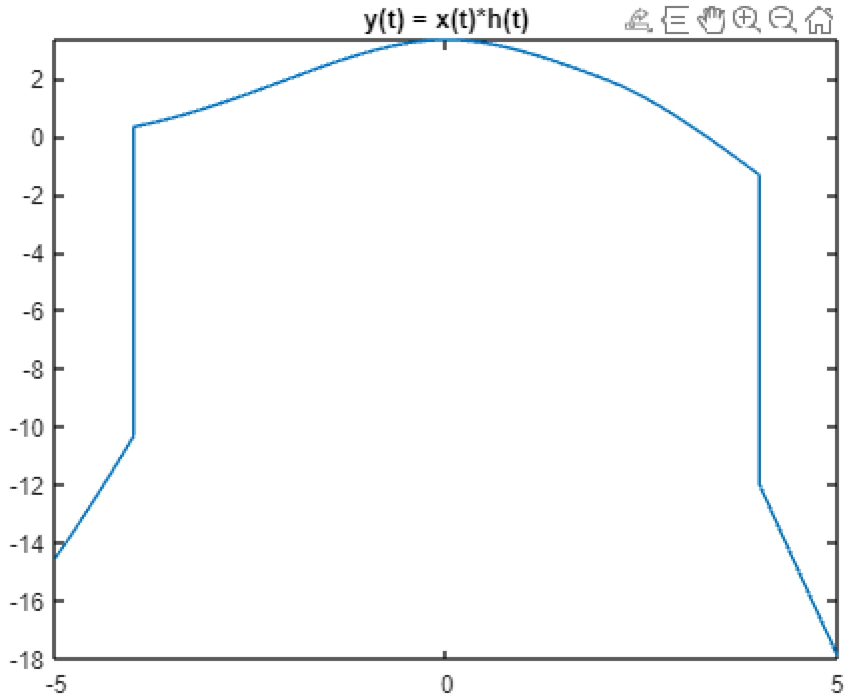
  Code:

```
syms w u(t) x(t) r(t) h(t) y(t)
u(t) = heaviside(t);
x(t) =((-0.5*t+2).*(u(t)-u(t-4)))+((0.5*t+2).*(u(-t)-u(-t-4)));
h(t) = ((-0.5*t+1).*(u(t)-u(t-2)))+((0.5*t+1).*(u(-t)-u(-t-2)));
y(t) = int(x(w)*h(t-w), w, -inf, inf);
fplot(x(t))
title('((-0.5*t+2).*(u(t)-u(t-4)))+((0.5*t+2).*(u(-t)-u(-t-4)))')
fplot(h(t))
title('((-0.5*t+1).*(u(t)-u(t-2)))+((0.5*t+1).*(u(-t)-u(-t-2)))')
fplot(y(t))
title('y(t) = x(t)*h(t)')
```

Output:







Explanation:

In this MATLAB code, symbolic functions are utilized to analyze a signal processing scenario. The step function u(t) is defined, and two functions, x(t) and h(t), are created. The function x(t) represents a piecewise linear signal composed of two segments: (-0.5t+2) for t between 0 and 4, and (0.5t+2) for t between -4 and 0. The function h(t) is also a piecewise linear signal with two segments: (-0.5t+1) for t between 0 and 2, and (0.5t+1) for t between -2 and 0. The code then calculates the convolution of x(t) and h(t), storing the result in y(t). The fplot function is employed to graphically represent x(t), h(t), and y(t), each with their respective titles. The first plot shows the piecewise linear signal x(t), the second plot illustrates the piecewise linear signal h(t), and the third plot displays the convolution result y(t) of x(t) and h(t).

Com. Sys. Lab 1 Rubric

Method of Evaluation: Executable code, Report submitted by students **Measured Learning Outcomes**:

CLO2: Develop software simulations to observe the performance of analog and digital communications systems.

CLO4: Report desired results proofs and calculations.

	Excellent 10	Good 9-7	Satisfactory 6-4	Unsatisfactory 3-1	Poor 0	Marks Obtained
Code (CLO2)	Correct code, easily understandable with comments where necessary	Correct code but without proper indentation or comments	Slightly incorrect code with proper comments	Incorrect code with improper format and no comments	Code not submitted	
Output (CLO2)	Output correctly shown with all Figures/ Plots displayed as required and properly labelled	Most Output/ Figures/ Plots displayed with proper labels	Some Output/ Figures/ Plots displayed with proper labels OR Most Output/ Figures/ Plots displayed but without proper labels	Most of the required Output/ Figures/ Plots not displayed	Output/ Figures/ Plots not displayed	
Answers (CLO2)	Meaningful answers to all questions. Answers show the understanding of the student.	Meaningful answers to most questions.	Some correct/ meaningful answers with some irrelevant ones	Answers not understandable/ not relevant to questions	Wrong Answers	
Lab Report (CLO4)	Report submitted with proper grammar and punctuation with proper conclusions drawn and good formatting	Report submitted with proper conclusions drawn with good formatting but some grammar mistakes OR proper grammar but not very good formatting	Some correct/ meaningful conclusions. Some parts of the document not properly formatted or some grammar mistakes	Conclusions not based on results. Bad formatting with no proper grammar/ punctuation	Report not submitted	
Total						